

REMARKS

Claims 1-59 are pending in the present application. Claims 1-4, 6, 8, 10, 20, 24, 29, 34, 39, 43, 44, 48, and 57-59 were amended. Reconsideration of the claims is respectfully requested.

In particular, claims 8, 43, and 46 were amended to correct grammatical errors. Claims 1, 6, 10, 20, 24, 29, 34, 39, 44, 48, and 57-59 were amended to recite the features of "receiving data describing objects in an original format, wherein the original format is unusable by a set of processing environments, and wherein the data comprises a graphical user interface object hierarchy comprising a sequence of functional units for execution in a graphical user interface"; "responsive to receiving a request for an object from a particular processing environment within the set of processing environments, retrieving data corresponding to the object from the database"; and "translating the data corresponding to the object into a selected format usable by the particular processing environment". Support for these amendments may be found on pages 11-13, 15, and Figure 4 of the Specification. Claim 2 was amended to recite "wherein the object contains a functional unit presented as an action to a user in the particular processing environment via a graphical user interface console, wherein the action has a same look and feel for each of the set of processing environments". Support for this amendment may be found on page 12, lines 10-15 of the Specification. Claim 3 was amended to recite "wherein the database format corresponds to information obtained from the graphical user interface hierarchy". Support for this amendment may be found on page 17, lines 14-26 of the Specification. Claim 4 was amended to recite "wherein the data describes a relationship between an instance of an action object and an instance of a functional unit definition object". Support for this amendment may be found on page 16, line 11 to page 17, line 4 and Figure 6 of the Specification.

I. 35 U.S.C. § 103, Obviousness, Claims 1-59

The examiner has rejected claims 1-59 under 35 U.S.C. § 103(a) as being unpatentable over Walsh et al. (U.S. Patent No. 6,810,429) (hereinafter "*Walsh*") in view of Lindberg et al. (U.S. Patent No. 6,732,109) (hereinafter "*Lindberg*"). This rejection is respectfully traversed.

With regard to claims 1, 39, 57, the examiner states:

As to Claims 1, 39, and 57, Walsh et al. discloses a method in a data processing system, the method comprising:

receiving data (i.e. add. Col. 6, lines 40-46) describing objects (i.e. document. Col. 6, lines 40-46);

storing the data in a database format in a database (i.e. data source. Col. 6, lines 40-46);

responsive to receiving a request (i.e. query. Col. 6, lines 4-24) for an object (i.e. add. Col. 6, lines 40-46);

retrieving data corresponding to the object from the database (i.e. a relational database. Col. 6, lines 25-29); and

translating the data corresponding to the object into a form for use by the processing environment (i.e. in the form of a XML document Col. 6, lines 4-24).

Walsh et al. does not explicitly teach transferring data.

Lindberg et al. teaches transferring data (i.e. transferring information. Col. 2, lines 56-57).

Therefore, it would have been obvious to a person having ordinary skill in the art at the time the invention was made to have modified Walsh et al. with transferring data.

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to have modified Walsh et al. by the teaching of Lindberg et al. because providing transferring data allows flexibility handling of tenants conflicting needs as taught by Lindberg et al. (Col. 1, lines 31-39).

(*Office Action*, dated January 19, 2005, pages 2-3.)

The examiner bears the burden of establishing a *prima facie* case of obviousness based on the prior art when rejecting claims under 35 U.S.C. § 103. *In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992). For an invention to be *prima facie* obvious, the prior art must teach or suggest all claim limitations. *In re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974). Amended independent claim 1, which is representative of amended independent claims 6, 10, 20, 24, 29, 34, 39, 44, 48, and 57-59 with regard to similarly recited subject matter, reads as follows:

1. A method in a data processing system for transferring data, the method comprising:

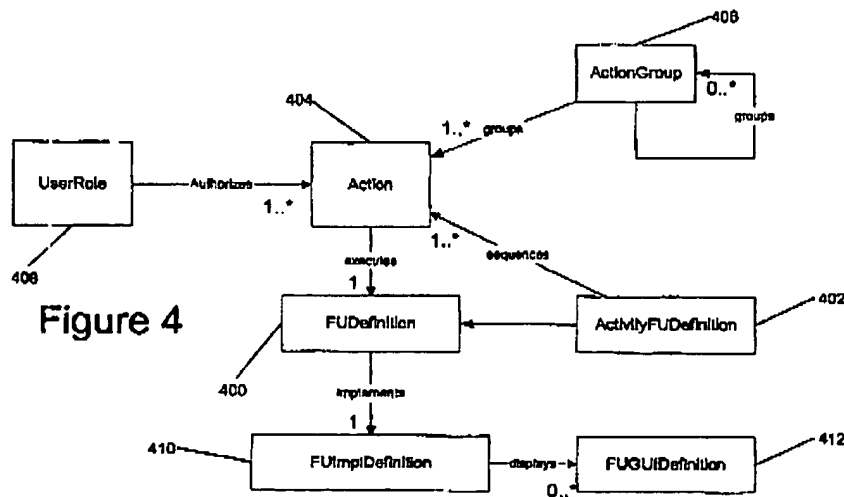
receiving data describing objects in an original format, wherein the original format is unusable by a set of processing environments, and wherein the data comprises a graphical user interface object hierarchy comprising a sequence of functional units for execution in a graphical user interface;

storing the data in a database format in a database;

responsive to receiving a request for an object from a particular processing environment within the set of processing environments, retrieving data corresponding to the object from the database; and

translating the data corresponding to the object into a selected format usable by the particular processing environment.

Amended claim 1 of the present invention recites receiving data describing objects in an original format, wherein the original format is unusable by a set of processing environments, and wherein the data comprises a graphical user interface object hierarchy comprising a sequence of functional units for execution in a graphical user interface. The present invention imports XML definitions of different objects that form the graphical user interface hierarchy. The object hierarchy also defines an execution sequence to a graphical user interface console. For example, Figure 4 of the present invention illustrates an example diagram of management components for defining an execution sequence to a graphical user interface console:



Activity functional unit 402 represents a sequence of work units making up an activity and identifies actions that form the execution sequence. Action 404 represents a functional unit 400

to the user and identifies a functional unit to execute. Action 404 is presented to the user via a graphical user interface console.

Amended claim 1 also recites "responsive to receiving a request for an object from a particular processing environment within the set of processing environments, retrieving data corresponding to the object from the database", and "translating the data corresponding to the object into a selected format usable by the particular processing environment". Claim 1 of the present invention allows for customizing external data describing a graphical user interface and providing the external data to different applications teams, such that the different teams are able to create graphical user interfaces that have the same look and feel, regardless of the processing environment.

The *Walsh* reference is directed toward integrating computer systems found in many types of enterprises. As discussed in the Abstract, *Walsh* provides an enterprise integration system coupled to a number of legacy data sources. These data sources use different formats and access methods. The enterprise integration system in *Walsh* employs a back-end interface configured to convert input data source information to input XML documents and to convert output XML documents to output data source information; a front-end interface configured to convert output XML documents to output HTML forms and the input HTML forms to the XML documents; a middle tier containing a rules engine and rules database; design tools to define the conversion and the XML documents; and mobile agents configured to communicate the XML documents over the networked system and process the XML documents according to the rules.

Walsh does not teach or suggest receiving data describing objects in an original format, wherein the original format is unusable by a set of processing environments, and wherein the data comprises a graphical user interface object hierarchy comprising a sequence of functional units for execution in a graphical user interface. The examiner states that *Walsh* teaches receiving data describing objects in the passage below:

Add

The "add" operation inserts a new document into the data source. The caller supplies the document in the form of a DOM Document object. The bridge service returns the id of the newly added document. In the case of a relational database, the add operation maps to a SQL INSERT INTO statement.

(*Walsh*, col. 6, lines 40-46).

The passage above recites supplying a document to a data source in the form of a DOM (document object model) document object. Although the passage teaches receiving a document in the form of a document object, the *Walsh* reference does not teach or suggest receiving data comprises a graphical user interface object hierarchy. Rather, the *Walsh* reference merely teaches a system for receiving an HTML input of enterprise information from a caller and allowing the caller to add, update, delete, browse, and search for documents in a legacy data source by converting the HTML input to XML documents and processing the documents (col. 3, lines 6-22; col. 4, lines 56-58). The caller may also obtain information from the legacy data source, as the data source “publishes” or “serves” XML documents containing enterprise information (col. 4, lines 54-56). The enterprise information in a data source that is accessible to the caller comprises transactional data of the enterprise, “such as customer accounts, purchase orders, work items, work lists, and the like” (col. 4, lines 61-63). This enterprise information in the legacy data source is presented to the caller by converting the XML document to an HTML form.

As *Walsh* merely teaches supplying enterprise information in a document in the form of a document object, *Walsh* makes no mention of having the data comprise a graphical user interface hierarchy. Rather, *Walsh* only teaches that users are provided with a graphical user interface which comprises a “thin” user interface – with simple interactivity that can easily be customized as the environment in the enterprise changes – that uses “forms” to allow users to view and modify the enterprise information (col. 7, lines 5-10). The graphical user interface is described in *Walsh* as a front-end interface for “user presentation and interaction” (col. 7, lines 4-5). *Walsh* describes how the data is provided to the users by simply stating that the XML documents are formatted into “a form suitable for users” by converting the XML document to a HTML page using a style sheet 126, e.g. XSL, JSP or some other data replacement technique” (col. 7, lines 38-42). Thus, *Walsh* teaches accessing and updating data in legacy data sources and providing an interface to present the data to users and allow the users to interact with the data sources, and merely converting the data into a “form suitable for users” (e.g., HTML). However, *Walsh* fails to teach or suggest receiving data describing objects in an original format, wherein the data comprises a graphical user interface object hierarchy.

Furthermore, *Walsh* does not teach or suggest a graphical user interface object hierarchy comprising a sequence of functional units for execution in a graphical user interface. As *Walsh* does not teach or suggest receiving data comprising a graphical user interface object hierarchy, it must follow that *Walsh* also fails to teach or suggest a graphical user interface object hierarchy comprising a sequence of functional units for execution in a graphical user interface.

In addition, *Walsh* does not teach or suggest receiving data describing objects in an original format, wherein the original format is unusable by a set of processing environments, and translating the data corresponding to the object into a selected format usable by a particular processing environment in the set of processing environments. The examiner states that *Walsh* teaches translating the data corresponding to the object into a form for use by the processing environment in the passage below:

The "query" operation retrieves a document from the data source. The caller supplies the id 104 of the document to fetch. The bridge service returns the specified information in the form of a XML document according to one of the standard programming models supported by W3C, for example, a DOM document object or a SAX document. DOM (Document Object Model), is a programming interface specification that specifies a tree which applications may then explore or modify. SAX is an event-based tool, more or less 'reading' the document to the application using a set of named methods to indicate document parts. SAX is typically used where efficiency and low overhead are paramount, while the DOM is used in cases where applications need random access to a stable tree of elements. The interface allows us to generate and modify XML documents as full-fledged objects. Such documents are able to have their contents and data "hidden" within the object, helping us to ensure control over who can manipulate the document. Document objects can carry object-oriented procedures called methods.

(*Walsh*, col. 6, lines 40-46).

As the passage above shows, *Walsh* teaches a query operation which retrieves a document from the legacy data source. When a caller supplies an ID of a document, the information is returned to the caller in the form of an XML document. For example, as shown in Figure 1b of *Walsh*, a caller may query legacy data using a front-end HTML form. The front-end interface converts the input HTML form to XML documents for user by the back-end system. The back-end interface converts input data source information to XML documents, and the front-end interface converts the XML document

to HTML form. However, this *Walsh* system does not receive data in a form that is unusable by the processing environment, nor does *Walsh* mention the desirability of doing so. The back-end system receives data in XML form and returns information in XML form. The caller sends requests in HTML or "a form suitable to users" as stated above, and the caller receives the response in HTML form. Thus, *Walsh* does not teach or suggest receiving data describing objects in an original format, wherein the original format is unusable by a set of processing environments, and translating the data corresponding to the object into a selected format usable by a particular processing environment in the set of processing environments.

Lindberg does not cure the deficiencies of *Walsh*. As discussed in the Abstract, *Lindberg* is directed toward a system for transferring information between a user interface and a database over a global information network. The *Lindberg* system teaches accessing a relational database by a user interface via a global information network. The relational database is accessed through four intermediate processing layers: an interaction layer, an application layer, a business object layer, and an information model layer. Each layer appears to the next layer as an XML document. The use of the four intermediate layers provides a separation of concern between the structure of information and the structure of its use for developers of applications of the information in the database. The separation of concerns to the individual layers creates the ability to deliver functionality to multiple tenants in the same system while allowing for the modification of necessary logic at each level independently with virtually all changes occurring within a layer, thereby localizing change and enabling the introduction of new or modified functionality in a controlled fashion (col. 5, lines 38-48).

Lindberg does not teach or suggest receiving data describing objects in an original format, wherein the original format is unusable by a set of processing environments, and wherein the data comprises a graphical user interface object hierarchy comprising a sequence of functional units for execution in a graphical user interface. Rather, *Lindberg* provides a user interface, which deals with data presentation and user interaction. The user interface itself is an XML or HTML document (col. 6, lines 57-59). The interaction layer defines how the user interface gains access to and communicates with the user interface/channel specific rendering (col. 22-25), and the application objects layer

comprises application objects which contain the logic that defines the processing behind graphical user interface buttons (col. 6, lines 27-34). The interaction layer includes a plurality of interaction objects. Each interaction object in *Lindberg* includes information identifying an application object and information specifying a manner in which requests from a user interface are to be processed for transmission to the application objects, and in which responses from the application object are to be transmitted to the user interface (col. 4, line 64 to col. 5, line 3). Thus, *Lindberg* merely teaches transferring content from layer to layer without shared processing, and rendering the information in the database on a user interface in an XML or HTML form. However, there is no mention in *Lindberg* of receiving data describing objects in an original format, wherein the original format is unusable by a set of processing environments, nor does *Lindberg* mention a hierarchy of graphical user interface objects comprising a sequence of functional units for execution in the graphical user interface. Consequently, *Lindberg* fails to teach or suggest receiving data describing objects in an original format, wherein the original format is unusable by a set of processing environments, and wherein the data comprises a graphical user interface object hierarchy comprising a sequence of functional units for execution in a graphical user interface, as recited in claim 1 of the present invention.

In addition, *Lindberg* does not teach or suggest receiving data describing objects in an original format, wherein the original format is unusable by a set of processing environments, and translating the data corresponding to the object into a selected format usable by a particular processing environment in the set of processing environments. *Lindberg* is not concerned with providing processing environments that are unable to use data in one format with the data in a format that the processing environment is able to utilize, and thus does not teach or suggest such a feature. Thus, *Lindberg* fails to teach or suggest receiving data describing objects in an original format, wherein the original format is unusable by a set of processing environments, and translating the data corresponding to the object into a selected format usable by a particular processing environment in the set of processing environments.

Furthermore, the examiner states "it would have been obvious to a person having ordinary skill in the art at the time the invention was made to have modified Walsh et al. by the teaching of Lindberg et al. because providing transferring data allows flexibly

handling of tenants conflicting needs as taught by Lindberg et al.” (*Office Action*, page 3). However, there is no teaching or suggestion in the references as to the desirability of including the features from the other references. As the examiner has failed to demonstrate any motivation or incentive in the prior art to combine and modify the references so as to achieve the claimed invention, the alleged combination can only be the result of impermissible hindsight reconstruction using applicant’s own disclosure as a guide. While applicant understands that all examination entails some measure of hindsight, when the rejection is based completely on hindsight, as in the present case, to the exclusion of what can be gleaned from the references, then the rejection is improper and should be withdrawn.

Even if *Lindberg* were combinable with *Walsh*, the result of such a combination would not be the invention as recited in claim 1. Rather, such an alleged combination would result in a system substantially as taught by *Walsh* in addition to the feature of transferring information between a user interface and a database. Even considering *Lindberg*, the cited references still fail to teach or suggest receiving data describing objects, wherein the data comprises a graphical user interface object hierarchy, and wherein the graphical user interface object hierarchy comprises a sequence of functional units for execution in a graphical user interface, as recited in claim 1 of the present invention.

In view of the above, applicant submits that independent claims 1, 6, 10, 20, 24, 29, 34, 39, 44, 48, and 57-59 are not taught or suggested by the alleged combination of *Walsh* and *Lindberg*.

Claims 2-5, 7-9, 11-19, 21-23, 25-28, 30-33, 35-38, 40-43, 45-47, and 49-56 are dependent claims depending on claims 1, 6, 10, 20, 24, 29, 34, 39, 44, and 48, respectively. At least by virtue of their dependency on claims 1, 6, 10, 20, 24, 29, 34, 39, 44, 48, these dependent claims are also patentable over *Walsh* and *Lindberg*. In addition, these claims include additional features not present in the *Walsh* and *Lindberg* references.

For example, claim 2 recites “wherein the object contains a functional unit presented as an action to a user in the particular processing environment via a graphical user interface console, wherein the action has a same look and feel for each of the set of processing environments”. The present invention provides the ability to present a

consistent graphical user interface, such that the mismatched look and feel that can occur among different applications that are developed by different application teams is avoided. In this manner, customers may be able to view a suite of applications that, although developed by different application teams, still have the same look and feel. Neither *Walsh* nor *Lindberg* teach or suggest this feature.

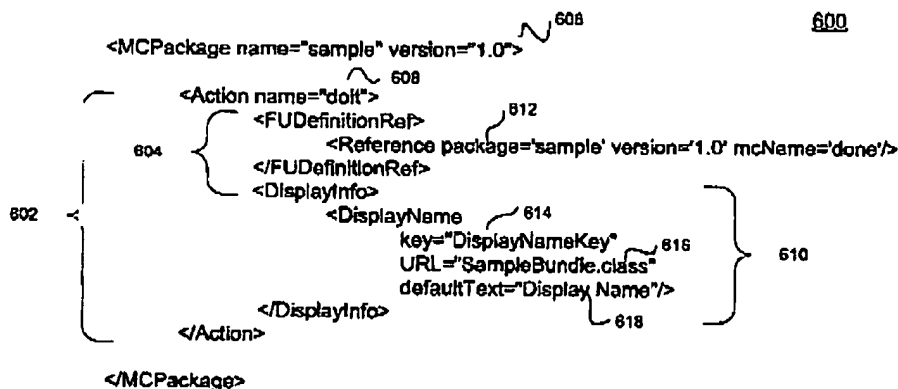
In another example, claim 3 of the present invention recites "wherein the database format corresponds to information obtained from the graphical user interface hierarchy". An illustration of an example database format in accordance with the present invention is shown in Figure 7 below:

Name of package 700	Version 702	Action 704	Name of function package 706	Version of function package 708	Name of function 710	Key 712	URL 714	Default text 716
Sample	1.0	Doit	Sample	1.0	Done	DisplayNameKey	SampleBundle.class	Display name

720

As neither *Walsh* nor *Lindberg* teach or suggest the graphical user interface hierarchy as described in claim 1, *Walsh* and *Lindberg* also fail to teach or suggest a database format that corresponds to information obtained from the graphical user interface hierarchy.

In addition, claim 4 of the present invention recites "wherein the data describes a relationship between an instance of an action object and an instance of a functional unit definition object". An illustration of an example relationship description in accordance with the present invention is shown in Figure 6 below:



Neither *Walsh* nor *Lindberg* teach or suggest data describing a relationship between an instance of an action object and an instance of a functional unit definition object.

Accordingly, applicant respectfully requests withdrawal of the rejection of claims 1-59 under 35 U.S.C. §103.

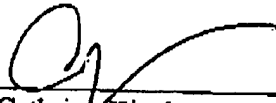
II. Conclusion

It is respectfully urged that the subject application is patentable over the cited references and is now in condition for allowance.

The examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: April 18, 2005

Respectfully submitted,



Cathrine Kinslow
Reg. No. 51,886
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777
Attorney for Applicant